When designing a complex system, there are many different server architectures for a system. Two major categories include:

[Monolithic Architecture]:

[Microservice Architecture]:

<table>
<thead>
<tr>
<th>Monolithic Architecture</th>
<th>Microservices Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Configuration and Deployment Challenges
One of the most challenging bits of microservices is managing the configuration and deployment of the microservices:
- What is the location of my dependencies?
- How do I quickly update the configuration?

Solution: ____________________________
Every process on every Operating System runs with a number of environmental variables.

<table>
<thead>
<tr>
<th>Command to List All Environment Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux: env</td>
</tr>
<tr>
<td>Windows PowerShell: dir env:</td>
</tr>
</tbody>
</table>

A few common ones:
- PATH:

- HOME (or HOMEPATH):

- USER (or USERNAME):

A few commonly defined in development environments:
- ENV:

- DEBUG:

- ...any number of custom application-specific ones...

Common Programming Convention: .env Files
A common, but not built-in, programming convention is to use .env files to specify deployment-specific environment variables.

<table>
<thead>
<tr>
<th>.env file</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLASK_RUN_PORT = 24000</td>
</tr>
</tbody>
</table>

...now, when we run Flask, we see it starts on a different port:

```
$ python3 -m flask run 
[...]
* Running on http://127.0.0.1:24000/ (Press CTRL+C to quit)
```

Networking Ports
Ports provide an application-specific connection allowing multiple services to run simultaneously on a single host.

Port Range:

Common Ports:

Reserved Ports:

Unreserved Ports:
**Isolation**

**Q:** Would you let your friend/roommate on your computer?

...system developers share the same concerns. There are many levels of increasing isolation provided by modern infrastructure:

1) **Process Isolation**

2) **User Account Isolation**

3) **Containerization**

4) **Virtualization**

<table>
<thead>
<tr>
<th>User Accounts</th>
<th>Containers</th>
<th>Virtualization</th>
</tr>
</thead>
</table>

**User Accounts**

Every user of modern systems has a “user account”, with a default “super user account” (called `root` on Linux).

Advantages:

Disadvantages:

**Containerization**

Containers rely on a “Container Engine” (ex: Docker) that allows apps to interact with the host system in an isolated way.

Advantages:

Disadvantages:
**Virtualization**

Virtual Machines (VMs) rely on a “Hypervisor” (ex: VMWare) that allows entire operating systems to run on a host system.

**Advantages:**

**Disadvantages:**

---

**Exploration of Different Use Cases:**

1) linux.ews.illinois.edu

2) Amazon AWS EC2

<table>
<thead>
<tr>
<th>vCPU</th>
<th>ECU</th>
<th>Memory (GiB)</th>
<th>Instance Storage (GiB)</th>
<th>Linux/UNIX Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>t3.nano</td>
<td>2</td>
<td>Variable</td>
<td>0.5 GiB</td>
<td>EBS Only</td>
</tr>
<tr>
<td>t3.micro</td>
<td>2</td>
<td>Variable</td>
<td>1 GiB</td>
<td>EBS Only</td>
</tr>
<tr>
<td>t3.small</td>
<td>2</td>
<td>Variable</td>
<td>2 GiB</td>
<td>EBS Only</td>
</tr>
<tr>
<td>t3.medium</td>
<td>2</td>
<td>Variable</td>
<td>4 GiB</td>
<td>EBS Only</td>
</tr>
<tr>
<td>t3.large</td>
<td>2</td>
<td>Variable</td>
<td>8 GiB</td>
<td>EBS Only</td>
</tr>
</tbody>
</table>

3) PrairieLearn

4) Our Class App?